# A LOW-COST DATA-ACQUISITION SYSTEM

BY KIYOHISA OKAMURA AND KAMYAB AGHAI-TABRIZ

*A compromise between cost and quality, this system is adequate for many research projects*

COMMERCIALLY AVAILABLE data-acquisition systems are quite expensive. A decent system may cost as much as or more than the entire annual equipment budget of an engineering department at a small educational institution like ours. Our solution to this problem was to design and build our own system. A reasonable compromise between price and quality, our system includes a Commodore 64 computer, a video display, a disk drive, and some miscellaneous hardware for about $800. It has only 8-bit data acquisition, but you can design a 12-bit system by using one and one-half I/O (input/output) ports (i.e., 12 bits) as the data-input pins. Furthermore, during breaks between experiments, our system can provide you with entertainment. Have you ever heard of a data-acquisition system you can play Pac-Man on?

### HARDWARE

The circuit diagram to interface the real world to the Commodore 64 is shown in figure 1, and the corresponding hardware is shown in photo 1. For analog-to-digital (A/D) conversion, we use an 8-bit ADC0804. To multiplex the multichannel analog input signals, we use the multiplexer (MUX) chip 4051. The outputs are connected to data lines PB0–PB7 of Complex Interface Adapter 2 (CIA2) through the Commodore 64's User Port CN2. The input channel selection is done by the three bits PB0, PB1, and PB2 of CIA1, which are connected respectively to C(MSB), B, and A(LSB) of the 4051. For example, channel 0 is selected by CBA-000, channel 1 by CBA-001, and so on. This multiplexing arrangement can accept up to eight analog signals. However, our plotting software is limited to three channels. The graphic resolution decreases as the number of channels displayed on the screen increases. Handshaking between the ADC and CN2 can be done through a pair of connections: $\overline{WR}$(ADC) to $\overline{PC2}$(Commodore 64) and $\overline{INT}$(ADC) to $\overline{FLAG}$(Commodore 64). The latter is optional, and we don't use it in our software.

The analog signal to be connected to each input terminal of the MUX CD4051 in figure 1 should be properly conditioned, which involves amplifying and biasing the signal so that the voltage level is between 0 and +5 V, because +5 V is used as a voltage reference in the ADC. The signal should be made to come as close as possible to the full range of the ADC, without exceeding the full-range limit, for maximum resolution. Therefore, you may need an amplifier between each transducer and the MUX. In our case, since the output of each transducer was relatively large, we used an analog computer for signal conditioning. For a very small signal you can use a differential amplifier. According to figure 1, one of the two lead wires for the input signal is for return and should be grounded.

*Kiyohisa Okamura, an associate professor of mechanical engineering and director of the Applied High-Tech Laboratory at North Dakota State University (Fargo, ND 58105), holds a Ph.D. from Purdue University. He is also a technical consultant for U.S.–Japanese biomedical engineering and computer businesses. Kamyab Aghai-Tabriz is a graduate student of mechanical engineering at North Dakota State University.*

*(continued)*

The ADC converts analog input voltage to 8-bit binary data with 0 V corresponding to 00000000 and +5 V to 11111111. The computer shows only the decimal equivalent on the screen, that is, 0 to 255 for 0 to 5 V, respectively. Any value between these two extremes is proportionally converted. For example, a converted data 1 (decimal unity) corresponds to an analog input to 0.02 V (1 × 5/255). Similarly, a data value of 37 corresponds to 0.73 V (37 × 5/255), and so on. If you want to store or display the value of input directly expressed in voltage, all you have to do is divide the acquired data by 51 (255/5).

Using this method of conversion together with a manufacturer's calibration data sheet for a transducer, we can determine the correlation between the original physical quantity and the acquired data in the computer. Another method we often use is direct calibration.

The accuracy of the A/D conversion depends partly upon the accuracy and stability of the voltage supplied to REF/2 (pin 9). We used the reference voltage from the Commodore 64's 5-V power supply. Our measurement shows that this voltage is actually 4.98 V with a ripple component of less than 0.5 percent. It is quite stable and accurate enough for undergraduate experiments conducted in our laboratories. If you want greater accuracy, use a more reliable voltage reference for pin 9.

The serial data is output to pin M of CN2, which is connected to the coaxial cable as shown in figure 2. The other end of the cable is connected to the serial port of a receiving computer either directly or through a line driver/receiver, depending on the compatibility of the two computers' serial ports. For example, the Commodore 64 and TRS-80 we are using in our laboratories are not RS-232C–compatible. In the Commodore 64, binary state 1 corresponds to +5 V and binary 0 to 0 V at pin M. On the other hand, at the RS-232C terminal of the TRS-80, binary state 1 corresponds to 0 V and binary 0 to +12 V. Therefore, these two computers are incompatible in both voltage levels and polarity. This incompatibility can be resolved by line driver MC1488 as shown. If the receiving computer uses +12 V and −12 V with inverted polarity, you should connect point P to the receiving RS-232C. With noninverted polarity, use point Q instead.

We use a 500-foot coaxial cable to connect a Commodore 64 in one laboratory to a TRS-80 in another laboratory. We haven't noticed any voltage drop or noise at the receiving end.

## SOFTWARE

[*Editor's note: The program for data acquisition is available for downloading via BYTEnet Listings. The telephone number is (603) 924-9820.*] The main portion of the program uses several assembly-language subroutines that are loaded in machine-language form via BASIC DATA statements. When you load the program, the menu in photo 2 appears. The menu and software are self-explanatory, so we'll only discuss the software briefly. When downloading the program, eliminate all state-

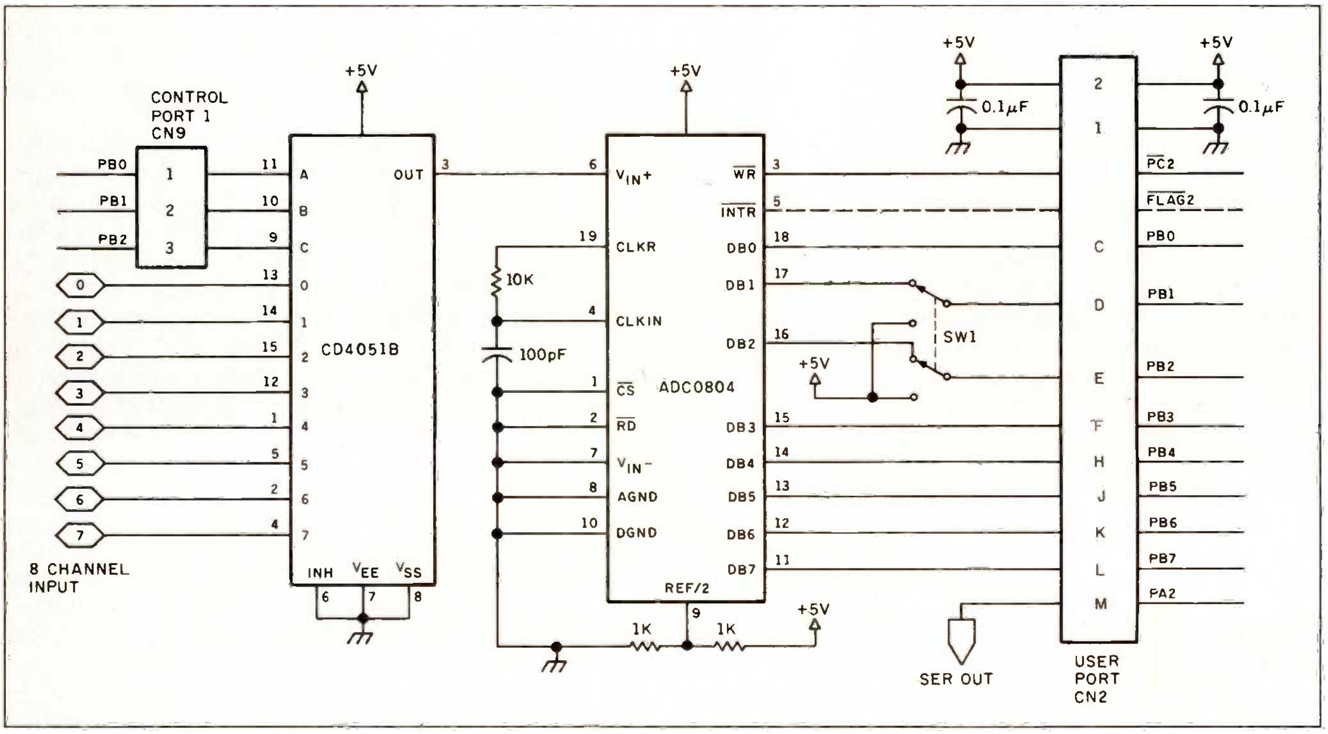


**Figure 1:** *A schematic for the A/D converter for the Commodore 64 data-acquisition system.*

ments headed with REM except for line 10, since they are strictly for comment and if typed in, they occupy too much space in RAM (random-access read/write memory).

When the main program is executed, all subroutines written in assembly language are poked into the appropriate locations as sequential data. Therefore, you should store the data (listings 2, 3, 4, and 5) as sequential files. Assign names (listing2, listing3, and so on) to these files. When the main program is executed, these programs will be poked into the locations shown in the first column of each listing.

A data-transmission subroutine is part of the main program. The transmission format is 2400 bps (bits per second), 7 data bits, 1 stop bit, and no parity check. This part of the program is also self-explanatory, but you have to remember to throw switch SW1 to the +5 V position when you use it. The screen displays the data as it is being transmitted from the Commodore 64. At the end of transmission, the screen displays an instruction: switch to ADC and press any key. You then throw SW1 back to the previous position so that the CIA is connected to the ADC.

The standard sampling rates of A/D conversion programmed in the main program are 1000, 500, and 100 samples per second; you can select the rate as part of the data-acquisition subroutine. In addition, you can set any sampling rate by yourself by adjusting parameters qq and ww in line 1110. This setting corresponds to the default value when the instruction for selecting the sampling rate is displayed on the screen. The maximum rate available is 4360 samples per second at ww = qq = 1. If you have three channels, this implies the sampling rate of 1453 samples/second for each channel. To lower the sampling rate, just increase qq and/or ww. These parameters are used in time-delay loops in the assembly program with parameter ww in the inner loop and parameter qq in the outer loop. Delay parameter ww has a greater effect on lowering the sampling rate than parameter qq does.

To calibrate the exact sampling rate, we used a square wave from a crystal oscillator as an input. Since the frequency of the crystal oscillator is quite accurately known, the sampling rate can therefore be determined.

(continued)

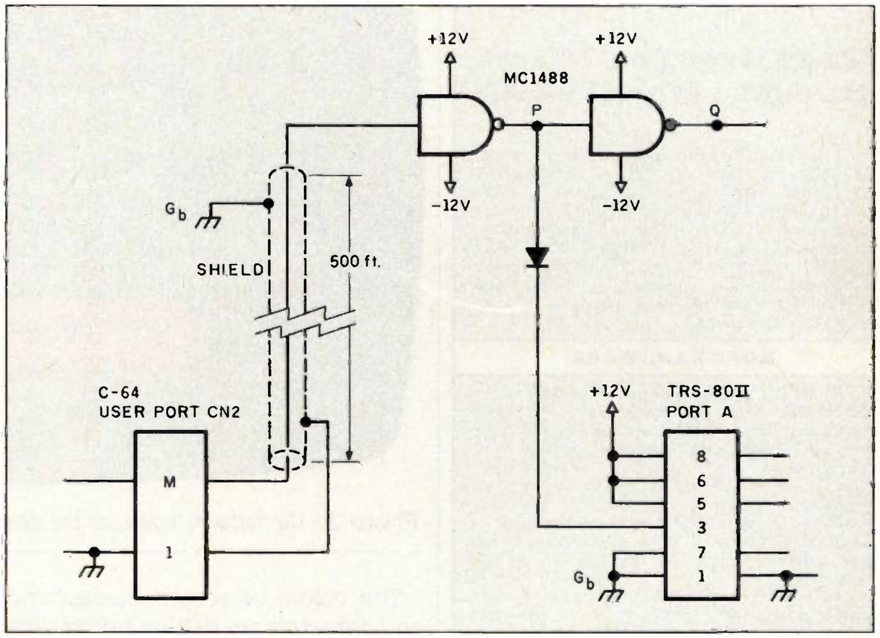


**Figure 2:** TTL *(transistor-transistor logic) to* RS-232C-*level conversion.*

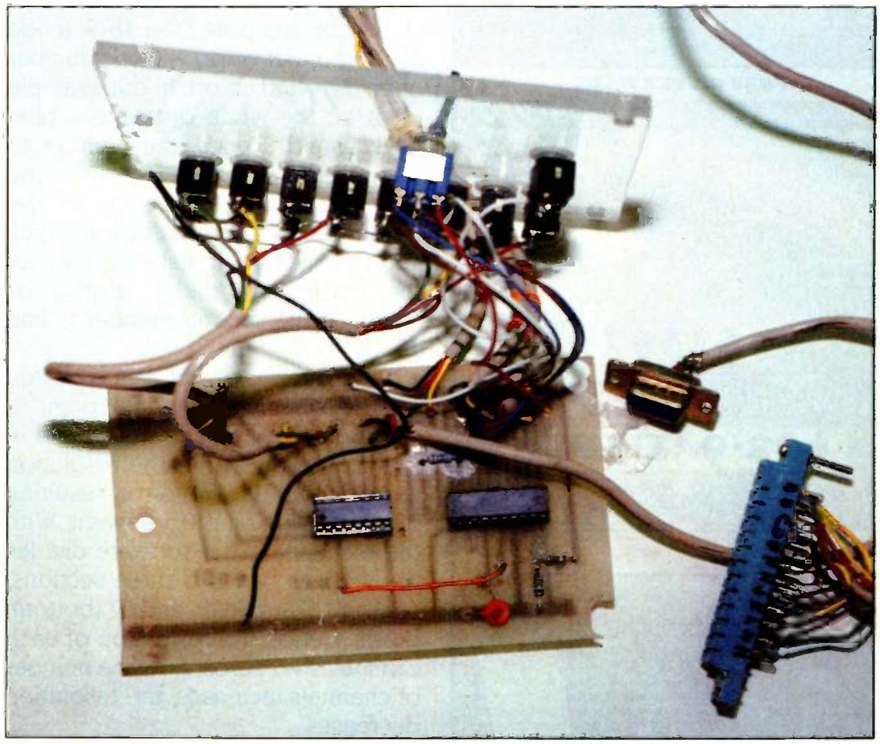

**Photo 1:** *The* A/D *converter.*


Photo 2: *The software menu for the data-acquisition system.*

The colors of screen background and data dots are determined by line 1470 in listing 1. You can change these colors by replacing the number 22 with another number. The number should be calculated as: 16 × (code number of dot color) + (code number of background color). In our example we used the white dots and a blue background. Hence, the number to be poked in is: $16 \times 1 + 6 = 22$. You can find the color codes in the Commodore 64 reference manual. You can also manipulate the color of the border in graphic display by changing the second number in line 1520.

When one channel of data is plotted on the screen, each data point is represented by one of 200 pixels in the vertical direction. The resolution represented by the error resulting from bit mapping is 0.5 percent. With three channels, the software divides the vertical axis into three sections: 66 (top), 67 (middle), and 67 (bottom) pixels. Hence, the resolution of each channel is 1.5 percent. As the number of channels increases, the resolution decreases.

The program stores data sequentially in RAM. In case of multiple channels (e.g., displacement $x$ for channel 0, velocity $v$ for channel 1, and acceleration $a$ for channel 2) the data is stored in the following order: $x(1)$, $v(1)$, $a(1)$, $x(2)$, $v(2)$, $a(2)$, $x(3)$, . . . , where $x(1)$ and $x(2)$ are the first and the second bytes of data for $x$, and so on. They are stored sequentially in RAM with the starting address of 32769. The number of data points for each channel is 320 by default but can be changed. Since there are 320 pixels in the horizontal direction of the screen, 320 data points per channel is the maximum number of data points that can be displayed at one time.

## CONCLUSION

We've found this system perfect for student use and adequate for some types of research. Though the system has many limitations, it is inexpensive and, above all, it's better than no system at all. ■

We would like to express our appreciation for the help Mr. William Welscher, a graduate student of agricultural engineering at North Dakota State University, gave us during the preparation of the manuscript of this article.